*Turkish Journal of Computer and Mathematics Education Vol.12 No.13(2021), 2436-2445*

*Research Article*

# Analysis of Various Visual SLAM Algorithms

**Dr S Mary Joans**
Department of Electronics and Communication Engineering
Velammal Engineering College
Chennai, India
ecehod@velammal.edu.in

**Vijay Logesh T S**
Department of Electronics and Communication Engineering
Velammal Engineering College
Chennai, India
vlogesh453@gmail.com

**P Vinayagam**
Department of Electronics and Communication Engineering
Velammal Engineering College
Chennai, India
vinayagam@velammal.edu.in

**Suriyanarayanan A**
Department of Electronics and Communication Engineering
Velammal Engineering College
Chennai, India
suriyashadows111@gmail.com

**Vancheeswaran Vaidyanathan**
Department of Electronics and Communication Engineering
Velammal Engineering College
Chennai, India
vigneshvanchy@gmail.com

**Saravanan A**
Department of Electronics and Communication Engineering
Velammal Engineering College
Chennai, India
saravananshaw151@gmail.com

**Article History**: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 4 June 2021

**Abstract—**Simultaneous Localization and Mapping (SLAM) is technique that is used to perform mapping and identifying the position of oneself in an unknown or unfamiliar region. This technique is being extensively investigated for application in areas such as autonomous vehicles, robotics, virtual reality and augmented reality. This paper first provides an overview of SLAM technique and explores a few existing visual SLAM algorithms (ORBSLAM2, ORBSLAM3 and DynaSLAM). After this, the performance of these algorithms on benchmarking datasets such as KITTI, TUM and EUROC is analyzed by considering parameters such as absolute and relative pose error. The plot of ground truth and estimated trajectory for these algorithms are also shown in the results section of the paper. The analysis was done using a virtual machine running Ubuntu 20.04 in AWS. A tabulation containing the results obtained from the evaluation tool is presented in the results section of the paper.

***Keywords****— SLAM, APE, RPE, visual SLAM*

## I. INTRODUCTION

The advent of technologies like autonomous vehicles [1][2], robotics [3][4], virtual and augmented reality [4][5][6] very often require some form of visual odometry or SLAM to perform their operations. Therefore, the research in this field has been very interesting especially in the past decade when several algorithms for SLAM were devised. As described before SLAM deals with creating a map of an unknown or unexplored environment and simultaneously knowing where the system is located in the environment. The need for SLAM arises due to the inaccuracies in GPS based methods which can result in issues while operating autonomous vehicles.

The basic steps involved in a SLAM framework include odometry, landmark prediction and extraction, matching of data, estimating the pose, and updating the maps [7]. These are the most fundamental steps in any SLAM algorithm, and they are often executed cyclically.

The basic steps often require a host of sensors for optimal operation, some of which are discussed here, in this paragraph. Acoustic sensors such as Sonar and Ultrasound have been tested and found to have some issues related to sensitivity and hence are not widely adopted [7]. The most popular sensor for SLAM is LIDAR based system. These systems are known for their accuracy and speed, and hence are widely used. However, these systems are very expensive and heavy and cannot be used for very small robots. Camera based systems are another choice for SLAM and these methods are called Visual SLAM techniques as they rely on the visual feed from the camera. There are three different versions in Visual SLAM, monocular, stereo and RGB-D. Monocular visual SLAM is very simple and is not highly accurate, stereo and RGB-D methods require more hardware setup than monocular SLAM but have better accuracy compared to monocular methods.

## II. OVERVIEW OF THE ALGORITHMS

We analyze 3 algorithms in this paper: ORBSLAM 2, ORBSLAM 3 and DynaSLAM. All these methods are visual SLAM algorithms. A brief overview of these algorithms is provided in this section.

### *A. ORBSLAM 2*

ORB-SLAM 2 is a SLAM algorithm that is used for real time performance on different camera configurations such as RGB-D, Monocular and Stereo [8]. It constructs the trajectory of the camera and a sparse reconstruction of the 3D environment. The algorithm performs loop closing, re-localization and map reuse. ORB-SLAM uses the monocular feature-based ORB-SLAM [11] for RGB-D and stereo configurations. Some of the prominent features of ORBSLAM 2 [8] are full Robot Operating System compatibility, out of box camera compatibility and map save and load capability. The system has three main parallel threads [], and they are:

1. The tracking [8] basically performs feature matching and minimises reprojection error.
2. The local mapping [8] for local map management and optimisation.
3. The loop closing [8] for large loops detection and reducing the accumulated drift by using pose-graph optimisation techniques.

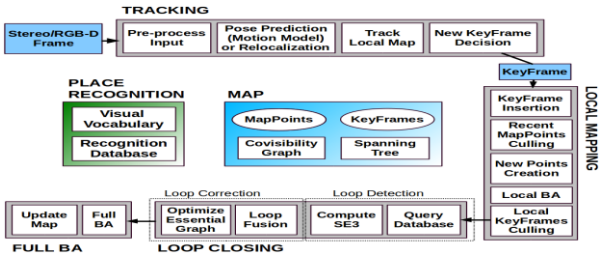


Fig. 1. Figure shows the overall block diagram of the ORBSLAM 2 algorithm with its various modules. [8]

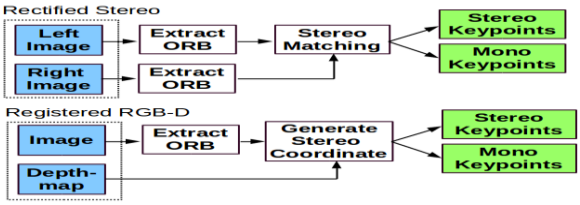


Fig. 2. Figure shows the input preprocessing step of ORBSLAM 2 [8]

Figure 1 and 2 show the overall block diagram and the input preprocessing of ORBSLAM 2 algorithm.

*B. ORBSLAM 3*

ORBSLAM 3 supports 3 different variations of SLAM, namely: visual SLAM, visual coupled with inertial data SLAM [10][12] and multiple map [10] SLAM with the camera configurations already mentioned for ORBSLAM 2 [9]. It allows using both pinhole and fisheye lens models [18][19]. It provides multiple map system that depends on new place recognition method with improved recall [9]. This algorithm is best suited for large, small and open-door navigation. ORBSLAM 3 can survive for a long period of time with poor visual information. This is achieved by creating a new map of the region which will be integrated with previously generated map while revisiting the mapped regions. The ORBSLAM 3 architecture is similar to the ORBSLAM 2 architecture, it provides certain additional modules such as: atlas [13], tracking thread, Local mapping thread and Loop and map merging thread [9]. Figure 3 shows the overall block diagram of ORBSLAM 3, this can be compared with Figure 2 for understanding the additional features added to ORBSLAM 2.

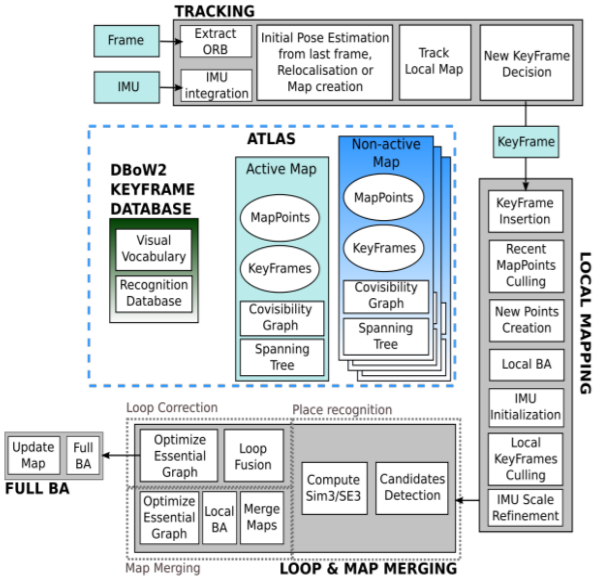


Fig. 3. Block Diagram of ORBSLAM 3 showing the various steps involved for performing the SLAM operation which have been built on top of ORBSLAM 2 [9].

*C. **DynaSLAM***

It is a visual SLAM algorithm that has been built over ORB-SLAM 2. Dynamic object detection and background inpainting are the two additional features provided by this algorithm [14]. DynaSLAM performs

better in dynamic scenarios for the various camera configurations such as stereo, RGB-D and monocular. It uses a combination of multi-view geometry and deep learning to enhance its performance [14].

Firstly, dynamic objects are detected by the use of a fully convolutional network called Mask R-CNN. With this technique all dynamic objects are detected a priori, but the moving objects are not detected. Secondly, dynamic objects are detected with a geometry method based on depth changes. With this method it is possible to detect the movement in the foreground of the scene. The combination of FCN and geometry approach results in optimal performance. Figure 4 shows the block diagram of DynaSLAM.

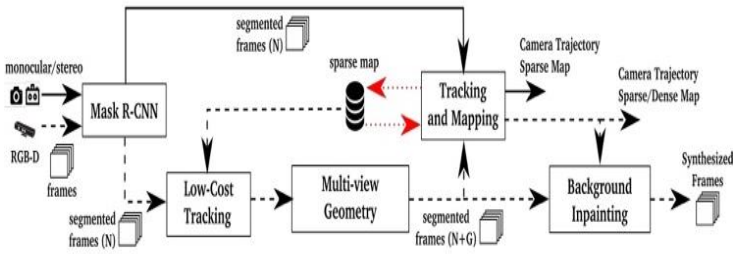


Fig. 4. Block Diagram of DynaSLAM showing the various steps involved for performing the SLAM operation [14].

## III. Evaluation Methodology

### *A. Hardware Setup*

We use a virtual machine in Amazon Web Servies for simulating the algorithms and for comparing the results produced by these algorithms. Listed below are the specification of the virtual machine:

- Ubuntu 20.04 (Operating System)
- 16 GB RAM
- 4 vCPU
- 1 NVIDIA T4 GPU (This GPU accelerates diverse cloud workloads, including deep learning training and interference and graphics).

### *B. Software Dependencies*

We used the following software dependencies to install the SLAM techniques on the VM: C++ 11, Pangolin for Visualization and UI, Open CV 3.2 for Manipulation and feature extraction, Eigen 3.3 [17] for Linear Algebra, DBoW 2 [15] for Place Recognition, g2o [16] for Non-Linear Optimizations and Python.

### *C. Benchmarking Datasets*

- KITTI Dataset:

  The KITTI dataset [20] is mainly applied for the benchmarking of outdoor environment recognition and navigation. It has stereo camera data, visual odometry data, optical flow data and data for 3D tracking and object detection. We are more interested in the visual odometry data for the analysis. This dataset consists of 22 sequences with ground truth data available for 11 sequences for evaluating SLAM algorithms. Development kits in C++ and MATLAB have also been provided for ease in accessing the data. The dataset has been collected using 2 colour and 2 grayscale video cameras, HDL-64E (Velodyne) 3D laser scanner and GPS/IMU unit with RTK correction.

- EuRoC Dataset:

  The EuRoC dataset [21] is mainly useful for the benchmarking of Indoor navigation and mapping. The dataset contains camera images (stereo), IMU measurements which are synchronized and the ground truth for each frame. It also provides the raw sensor data and a calibration dataset. The stereo data has been obtained using a global shutter camera (Aptina) and WVGA monochrome camera. The IMU data has been collected using a MEMS IMU sensor. Ground-Truth data has been obtained using a combination of motion capture system and 3D laser scan.

*D. Evaluation Package and Metric:*

We use evo [22] – a python package for the evaluation of the performance of the chosen algorithms. It supports handling, evaluating and comparing the trajectory outputs of odometry and SLAM algorithms. It supports previously introduced datasets like KITTI, EuRoC MAV, TUM Monocular datasets and ROS (Robot Operating System) bag files with certain message types. The built-in plotting provides excellent visuals and it also provides an implementation of SE Umeyama alignment and scale that is usually required for monocular SLAMs.

Evo can be used to compute RPE (Relative Pose Error) and APE (Absolute Pose Error). The absolute pose error is a metric for evaluating the global performance of a SLAM trajectory whereas, the relative pose error is a metric for evaluating the local performance of a SLAM trajectory. For more in-depth explanation of these terms one can refer to [23][24][25][26].

The flow chart for performing the evaluation of the algorithms is shown in Fig 5.

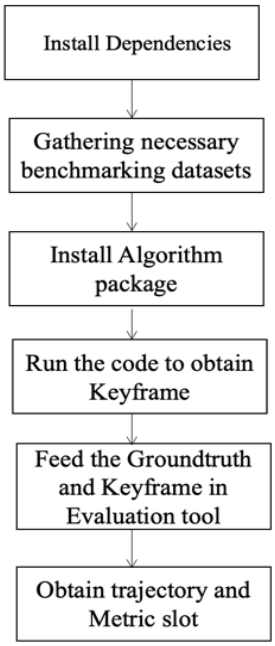


Fig. 5. Flow chart explains the steps involved in the evaluation process.

## IV. RESULTS

The trajectory estimated by ORBSLAM 2, ORBSLAM 3 and DynaSLAM were obtained by running the algorithms on EuRoC and KITTI datasets. This trajectory was compared with the ground truth using the evaluation package and the RPE and APE were obtained. We used 3 datasets from EuRoC dataset (MH01, MH03 and MH05) for getting this result. MH01 is the easiest dataset and MH03 is the hardest dataset in the EuRoC datasets that we used. EuRoC dataset is not supported by DynaSLAM in its current form, so only

ORBSLAM 2 and ORBSLAM 3 were analyzed with the EuRoC dataset. Apart from this, we used the 3rd, 6th and 7$^{th}$ sequences in the KITTI dataset. KITTI 6 and 7 sequences have slightly more dynamic objects than the 3 sequence. The RPE and APE tabulation (Table 1 and Table 2) and the trajectory (Fig 6 – Fig 11) obtained by running the algorithms along with the ground truth are shown in the following sections.

*A. Trajectory Plots*

Figure 5 to Figure 10 shows the trajectory plot for all the algorithms for both the KITTI and EuRoC dataset. Figure 6 to Figure 8 shows the KITTI trajectory for all three algorithms DynaSLAM (Blue), ORBSLAM 3 (Green) and ORBSLAM 2 (Red) along with the ground truth (dotted grey line). This can be used to visualize how each algorithm performs by seeing the amount of drift of each path from the ground truth. Similar visualization can be done for EuRoC dataset shown by Figure 9 to Figure 11. In this ORBSLAM 2 path is shown in green, ORBSLAM 3 path is shown in blue, and the ground truth is shown by grey dotted lines.

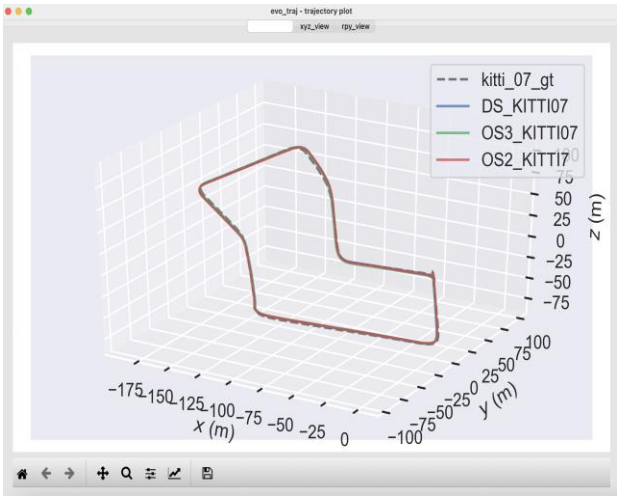


Fig. 6. KITTI 07 Trajectory for DynaSLAM, ORBSLAM 2 and ORBSLAM

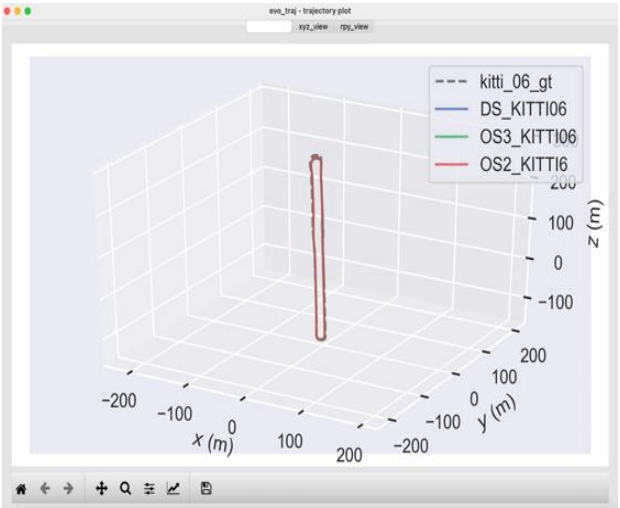


Fig. 7. KITTI 06 Trajectory for DynaSLAM, ORBSLAM 2 and ORBSLAM 3.

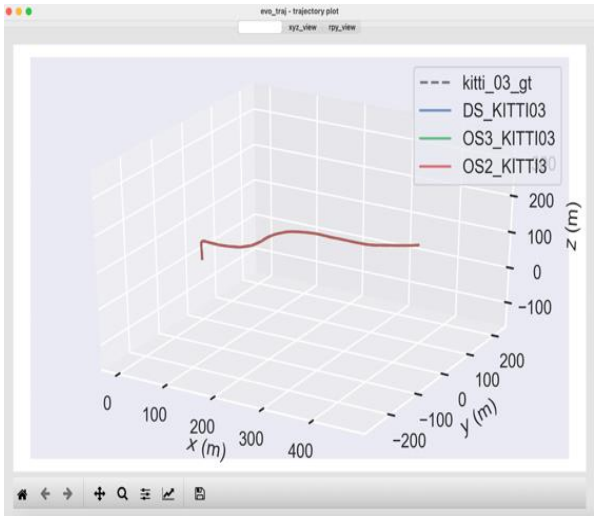


Fig. 8. KITTI 03 Trajectory for DynaSLAM, ORBSLAM 2 and ORBSLAM 3.

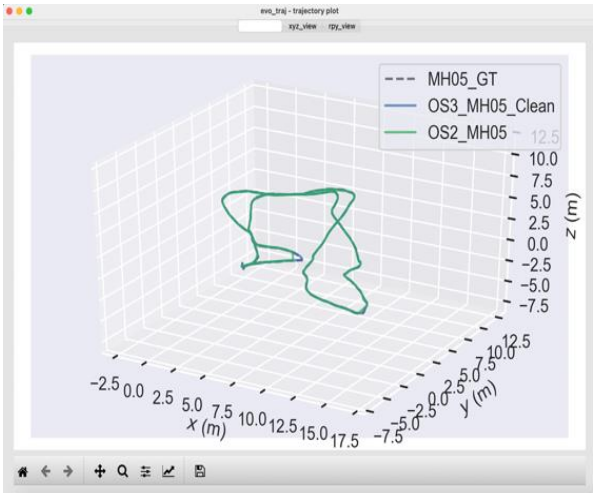


Fig. 9. MH 05 Trajectory for ORBSLAM 2 and ORBSLAM 3.

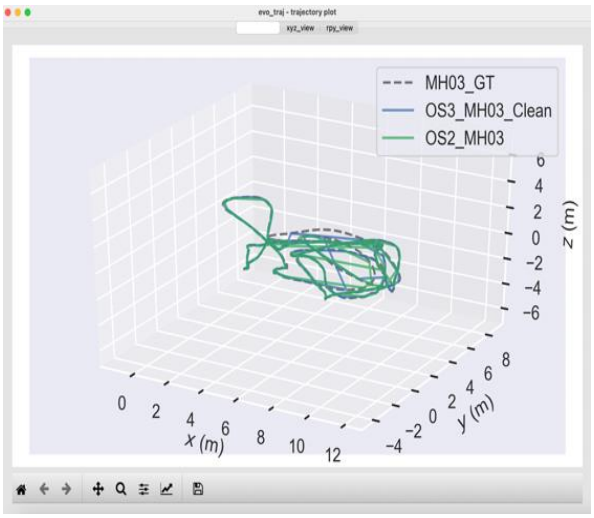


Fig. 10. MH 03 Trajectory for ORBSLAM 2 and ORBSLAM 3.

| Dataset | Algorithm | max | mean | median | min | rmse | sse | std |
|---|---|---|---|---|---|---|---|---|
| **MH01(Easy)** | ORBSLAM 2 | **0.073915** | **0.039276** | **0.032392** | **0.002869** | **0.045008** | **0.407177** | **0.021980** |
| | ORBSLAM 3 | 0.100175 | 0.044020 | 0.037867 | 0.005653 | 0.051226 | 0.650769 | 0.026197 |
| **MH03 (Med)** | ORBSLAM 2 | **0.081756** | **0.033444** | **0.029104** | 0.007115 | **0.037015** | **0.217842** | **0.015861** |
| | ORBSLAM 3 | 0.115592 | 0.044654 | 0.041981 | **0.005166** | 0.048893 | 0.499622 | 0.019914 |
| **MH05 (Hard)** | ORBSLAM 2 | 0.116224 | 0.044392 | 0.041578 | 0.013740 | 0.049095 | **0.472416** | 0.020968 |
| | ORBSLAM 3 | **0.101896** | **0.041247** | **0.037008** | **0.006401** | **0.044845** | 0.504782 | **0.017600** |
| **KITTI 03** | ORBSLAM 2 | 3.005655 | **0.989722** | **0.938982** | 0.244656 | **1.083792** | **317.143604** | 0.441651 |
| | ORBSLAM 3 | **2.754388** | 1.013182 | 0.942783 | **0.189780** | 1.090998 | 349.941518 | **0.404648** |
| | DynaSLAM | 3.893154 | 1.432407 | 1.358638 | 0.334895 | 1.581656 | 675.441778 | 0.670707 |
| **KITTI 06** | ORBSLAM 2 | 24.436146 | 13.137696 | 14.117423 | **0.222167** | **15.021108** | **88222.772567** | **7.282488** |
| | ORBSLAM 3 | 23.902323 | 13.288679 | 14.630621 | 0.408032 | 15.285470 | 89252.621936 | 7.553583 |
| | DynaSLAM | **23.424556** | **13.064751** | **13.649723** | 0.246729 | 15.088736 | 93800.024613 | 7.548659 |
| **KITTI 07** | ORBSLAM 2 | 4.397640 | 1.880293 | 1.803562 | 0.241955 | 2.175197 | 1883.129302 | 1.093608 |
| | ORBSLAM 3 | **3.525272** | **1.669764** | 1.595881 | **0.151303** | **1.832031** | **1282.121116** | **0.753808** |
| | DynaSLAM | 4.685647 | 1.865652 | **1.541146** | 0.332128 | 2.173077 | 1827.515432 | 1.114273 |

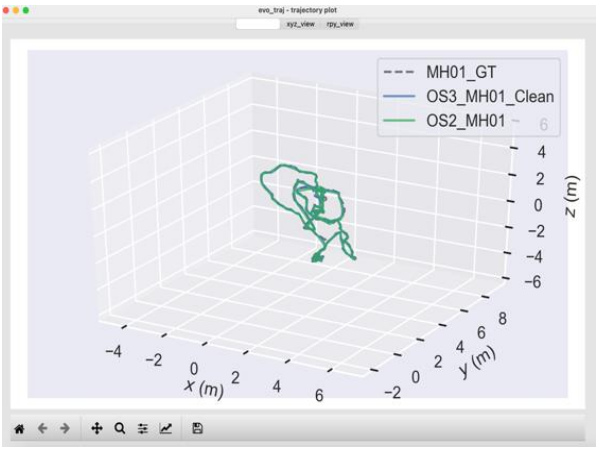


Fig. 11. MH 01 Trajectory for ORBSLAM 2 and ORBSLAM 3.

*B. APE and RPE Tabulation:*

TABLE I. THIS TABLE SHOWS THE APE METRICS

## V. CONCLUSION

After analyzing the APE and RPE table and trajectory plot of ORBSLAM 2, ORBSLAM 3 and DynaSLAM, it can be inferred that the performance of these algorithms is fairly very close, the differences between the error generated by one algorithm and another is very small. However, a general trend that can be seen is that ORBSLAM 3 performs better that ORBSLAM 2 in harder datasets. ORBSLAM 2 in other cases seems to perform better than DynaSLAM and ORBSLAM 3. However, the extra features supported by ORBSLAM 3 such as support for various lens models and others mentioned in the section 2.2 makes it an interesting candidate. DynaSLAM too, seems to be performing quite close and might be a good option in highly dynamic environments.

## REFERENCES

[1] S. Milz, G. Arbeiter, C. Witt, B. Abdallah and S. Yogamani, "Visual SLAM for Automated Driving: Exploring the Applications of Deep Learning," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Salt Lake City, UT, USA, 2018, pp. 360-36010, doi: 10.1109/CVPRW.2018.00062.

[2] G. Bresson, Z. Alsayed, L. Yu and S. Glaser, "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving," in *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194-220, Sept. 2017, doi: 10.1109/TIV.2017.2749181.

[3] Auat Cheein, F.A., Lopez, N., Soria, C.M. *et al.* SLAM algorithm applied to robotics assistance for navigation in unknown environments. *J NeuroEngineering Rehabil* **7,** 10 (2010). https://doi.org/10.1186/1743-0003-7-10

[4] Taketomi, T., Uchiyama, H. & Ikeda, S. Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ T Comput Vis Appl* **9,** 16 (2017). https://doi.org/10.1186/s41074-017-0027-2

[5] F. Munoz-Montoya, M. -. Juan, M. Mendez-Lopez and C. Fidalgo, "Augmented Reality Based on SLAM to Assess Spatial Short-Term Memory," in *IEEE Access*, vol. 7, pp. 2453-2466, 2019, doi: 10.1109/ACCESS.2018.2886627.

[6] G. Reitmayr *et al.*, "Simultaneous Localization and Mapping for Augmented Reality," *2010 International Symposium on Ubiquitous Virtual Reality*, Gwangju, Korea (South), 2010, pp. 5-8, doi: 10.1109/ISUVR.2010.12.

[7] T.J. Chong, X.J. Tang, C.H. Leng, M. Yogeswaran, O.E. Ng, Y.Z. Chong, Sensor Technologies and Simultaneous Localization and Mapping (SLAM), Procedia Computer Science, Volume 76, 2015, Pages 174-179, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2015.12.336.

[8] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," in *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255-1262, Oct. 2017, doi: 10.1109/TRO.2017.2705103.

[9] Carlos Campos and Richard Elvira and Juan J. Gómez Rodríguez and José M. M. Montiel and Juan D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM", 2020, 2007.11898, arXiv.

[10] Raúl Mur-Artal, and Juan D. Tardós, Visual-inertial monocular SLAM with map reuse, IEEE Robotics and Automation Letters, vol. 2 no. 2, pp. 796-803, 2017.

[11] Raúl Mur-Artal, José M. M. Montiel and Juan D. Tardós. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147-1163, 2015.

[12] Carlos Campos, J. M. M. Montiel and Juan D. Tardós, Inertial-Only Optimization for Visual-Inertial Initialization, *ICRA 2020*.

[13] Richard Elvira, J. M. M. Montiel and Juan D. Tardós, ORBSLAM-Atlas: a robust and accurate multi-map system, *IROS 2019*

[14] B. Bescos, J. M. Fácil, J. Civera and J. Neira, "DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes," in *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076-4083, Oct. 2018, doi: 10.1109/LRA.2018.2860039.

[15] D. Galvez-López and J. D. Tardos, "Bags of Binary Words for Fast Place Recognition in Image Sequences," in *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188-1197, Oct. 2012, doi: 10.1109/TRO.2012.2197158.

[16] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige and W. Burgard, "G2o: A general framework for graph optimization," *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 3607-3613, doi: 10.1109/ICRA.2011.5979949.

[17] Gael Guennebaud and Benoit Jacob and others, Eigen v3, https://eigen.tuxfamily.org, 2010

[18] Sturm P. (2014) Pinhole Camera Model. In: Ikeuchi K. (eds) Computer Vision. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-31439-6_472

[19] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 8, pp. 1335-1340, Aug. 2006, doi: 10.1109/TPAMI.2006.153.

[20] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, 2012, pp. 3354-3361, doi: 10.1109/CVPR.2012.6248074.

[21] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. Achtelik and R. Siegwart, *The EuRoC micro aerial vehicle datasets*, International Journal of Robotic Research**,** DOI: 10.1177/0278364915620033, early 2016.

[22] Grupp, Michael, evo: Python package for the evaluation of odometry and SLAM, https://github.com/MichaelGrupp/evo, 2017

[23] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387–407, 2009.

[24] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.

[25] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. *In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012.

[26] Umeyama, Shinji. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 4:376-380, 1991.